STAC67: Regression Analysis

Lecture 19

Thibault Randrianarisoa

November 18, 2026

University of Toronto Scarborough

Review of Friday

- With $\ln Y$ as the response, one more unit of X_k multiplies the median of Y by e^{b_k}. With $\ln X$ as a predictor, b is read for 1% more X.
- R^2_p never decreases when a predictor is added: it cannot compare models of different sizes.
- Adjusted R^2, C_p, AIC, BIC and PRESS each penalize more parameters. A small C_p, close to p: little bias.
- Surgical unit, the four scores: every criterion chooses X_1, X_2, X_3.

Today: the end of model selection, the search procedures, then validation.

Model selection (continued)

The criteria at a glance

$MSE = \frac{SSE}{n-p}$

Criterion	A good model has	Price of one more parameter
R_p^2	(a large value)	none: cannot compare sizes
$R_{a,p}^2$	the largest value	one error degree of freedom ($\lvert t^* \rvert > 1$)
C_p	a small value, close to p	2, in units of σ^2
AIC_p	the smallest value	2
BIC_p	the smallest value	$\ln n$
$PRESS_p$	the smallest value	implicit: fitting noise predicts worse

The criteria at a glance

Criterion	A good model has	Price of one more parameter
R_p^2	(a large value)	none: cannot compare sizes
$R_{a,p}^2$	the largest value	one error degree of freedom ($\lvert t^* \rvert > 1$)
C_p	a small value, close to p	2, in units of σ^2
AIC_p	the smallest value	2
BIC_p	the smallest value	$\ln n$
$PRESS_p$	the smallest value	implicit: fitting noise predicts worse

All of them compare models for the same response on the same cases; not a model for Y with a model for $\ln Y$, and not models fitted on different rows.

Before comparison, either delete all missing values or [impute them]

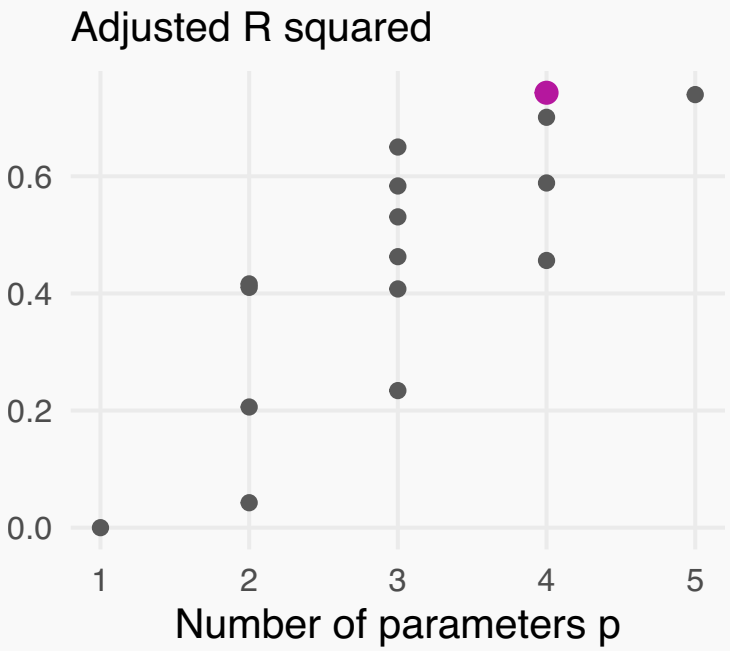
All 16 models

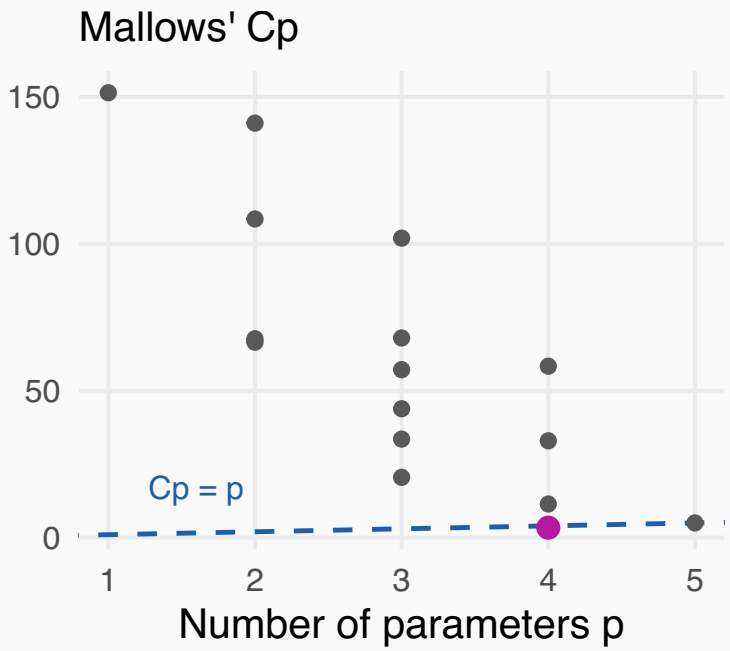
The criteria for every subset of X_1, X_2, X_3, X_4, with the best value of each criterion in colour:

Predictors	p	R_p^2	$R_{a,p}^2$	C_p	AIC_p	BIC_p	$PRESS_p$
none	1	0.000	0.000	151.44	−75.72	−73.73	13.292
X_1	2	0.061	0.043	141.09	−77.10	−73.12	13.508
X_2	2	0.221	0.206	108.47	−87.21	−83.23	10.738
X_3	2	0.427	0.416	66.52	−103.81	−99.83	8.329
X_4	2	0.421	0.410	67.70	−103.27	−99.29	8.024
$X_1\ X_2$	3	0.263	0.234	101.94	−88.19	−82.23	11.055
$X_1\ X_3$	3	0.548	0.531	43.87	−114.64	−108.68	6.989
$X_1\ X_4$	3	0.430	0.408	67.96	−102.07	−96.10	8.472
$X_2\ X_3$	3	0.663	0.650	20.52	−130.48	−124.51	5.066
$X_2\ X_4$	3	0.483	0.463	57.17	−107.34	−101.38	7.474
$X_3\ X_4$	3	0.599	0.583	33.54	−121.09	−115.12	6.123
$X_1\ X_2\ X_3$	4	0.757	0.743	3.39	−146.16	−138.21	3.914
$X_1\ X_2\ X_4$	4	0.487	0.456	58.36	−105.76	−97.81	7.902
$X_1\ X_3\ X_4$	4	0.612	0.589	32.96	−120.82	−112.87	6.209
$X_2\ X_3\ X_4$	4	0.718	0.701	11.43	−138.01	−130.05	4.598
$X_1\ X_2\ X_3\ X_4$	5	0.759	0.739	5.00	−144.59	−134.64	4.069

For a fixed value of p, all the criteria are monotone functions of SSE

Seeing the 16 models





Each point is a model; X_1, X_2, X_3 is in colour. Within a size, every criterion ranks the models as SSE_p does, except PRESS. The criteria matter when sizes are compared.

In R

For one candidate model, and the model with all the predictors:

```
fit  <- lm(log(Y) ~ X1 + X2 + X3, data = surg)
full <- lm(log(Y) ~ X1 + X2 + X3 + X4, data = surg)
n <- nrow(surg); p <- length(coef(fit))
summary(fit)$adj.r.squared                    # adjusted R2
sum(resid(fit)^2) / sigma(full)^2 - (n - 2 * p)    # Cp
extractAIC(fit)                               # p and AIC
extractAIC(fit, k = log(n))                   # p and BIC
sum((resid(fit) / (1 - hatvalues(fit)))^2)    # PRESS
```

In R

For one candidate model, and the model with all the predictors:

```
fit  <- lm(log(Y) ~ X1 + X2 + X3, data = surg)
full <- lm(log(Y) ~ X1 + X2 + X3 + X4, data = surg)
n <- nrow(surg); p <- length(coef(fit))
summary(fit)$adj.r.squared                    # adjusted R2
sum(resid(fit)^2) / sigma(full)^2 - (n - 2 * p)    # Cp
extractAIC(fit)                               # p and AIC
extractAIC(fit, k = log(n))                   # p and BIC
sum((resid(fit) / (1 - hatvalues(fit)))^2)    # PRESS
```

$$\text{PRESS} = \sum_{i=1}^{n} \left(\frac{e_i}{1 - h_{ii}} \right)^2$$

This gives 0.743, 3.39, -146.16, -138.21 and 3.914: the row of the table.

In R

For one candidate model, and the model with all the predictors:

```
fit  <- lm(log(Y) ~ X1 + X2 + X3, data = surg)
full <- lm(log(Y) ~ X1 + X2 + X3 + X4, data = surg)
n <- nrow(surg); p <- length(coef(fit))
summary(fit)$adj.r.squared                    # adjusted R2
sum(resid(fit)^2) / sigma(full)^2 - (n - 2 * p)    # Cp
extractAIC(fit)                               # p and AIC
extractAIC(fit, k = log(n))                   # p and BIC
sum((resid(fit) / (1 - hatvalues(fit)))^2)    # PRESS
```

This gives 0.743, 3.39, -146.16, -138.21 and 3.914: the row of the table.

`sigma(full)` is $\sqrt{MSE}$ of the full model; `hatvalues(fit)` gives the leverages h_{ii}.

With the eight predictors: 256 models

The best model of each size (smallest SSE_p), with the best value of each criterion over the 256 models in colour:

Predictors X_k	p	$R^2_{a,p}$	C_p	AIC_p	BIC_p	$PRESS_p$
3	2	0.416	117.48	−103.81	−99.83	8.329
2, 3	3	0.650	50.49	−130.48	−124.51	5.066
2, 3, 8	4	0.765	18.90	−151.00	−143.05	3.468
1, 2, 3, 8	5	0.816	5.73	−163.38	−153.43	2.736
1, 2, 3, 6, 8	6	0.821	5.53	−163.83	−151.89	2.782
1, 2, 3, 5, 6, 8	7	0.823	5.77	−163.86	−149.94	2.771
1, 2, 3, 5, 6, 7, 8	8	0.823	7.03	−162.74	−146.83	2.808
1, 2, 3, 4, 5, 6, 7, 8	9	0.819	9.00	−160.78	−142.88	2.931

With the eight predictors: 256 models

The best model of each size (smallest SSE_p), with the best value of each criterion over the 256 models in colour:

Predictors X_k	p	$R^2_{a,p}$	C_p	AIC_p	BIC_p	$PRESS_p$
3	2	0.416	117.48	−103.81	−99.83	8.329
2, 3	3	0.650	50.49	−130.48	−124.51	5.066
2, 3, 8	4	0.765	18.90	−151.00	−143.05	3.468
1, 2, 3, 8	5	0.816	5.73	−163.38	−153.43	2.736
1, 2, 3, 6, 8	6	0.821	5.53	−163.83	−151.89	2.782
1, 2, 3, 5, 6, 8	7	0.823	5.77	−163.86	−149.94	2.771
1, 2, 3, 5, 6, 7, 8	8	0.823	7.03	−162.74	−146.83	2.808
1, 2, 3, 4, 5, 6, 7, 8	9	0.819	9.00	−160.78	−142.88	2.931

The criteria no longer agree: R^2_a and AIC keep 6 predictors, C_p keeps 5, BIC and PRESS keep 4. And the differences between these models are small.

In R: the best model of each size

The package `leaps` searches all the subsets:

```
library(leaps)
best <- regsubsets(log(Y) ~ X1 + X2 + X3 + X4 + X5 + X6 +
                     X7 + X8, data = surg, nvmax = 8)
s <- summary(best)
s$which         # the predictors of the best model of each size
s$adjr2; s$cp   # their adjusted R2 and Cp
```

In R: the best model of each size

The package `leaps` searches all the subsets:

```
library(leaps)
best <- regsubsets(log(Y) ~ X1 + X2 + X3 + X4 + X5 + X6 +
                     X7 + X8, data = surg, nvmax = 8)
s <- summary(best)
s$which         # the predictors of the best model of each size
s$adjr2; s$cp   # their adjusted R2 and Cp
```

- `nbest = 3` keeps the three best models of each size.
- `s$bic` is our BIC_p up to a constant: same ranking.

In R: the best model of each size

The package `leaps` searches all the subsets:

```
library(leaps)
best <- regsubsets(log(Y) ~ X1 + X2 + X3 + X4 + X5 + X6 +
                     X7 + X8, data = surg, nvmax = 8)
s <- summary(best)
s$which         # the predictors of the best model of each size
s$adjr2; s$cp   # their adjusted R2 and Cp
```

- `nbest = 3` keeps the three best models of each size.
- `s$bic` is our BIC_p up to a constant: same ranking.

⚠ `regsubsets` sees columns, not variables. It may keep one indicator of a factor without the other (here X_8 without X_7): allowed, but X_8 then compares severe use with none and moderate together. It may also keep a square or a product without its lower-order terms, which we do not allow (Lectures 16 and 17): add those terms back before using such a model.

What the criteria do not do

- They rank models, they do not check them. The residual analysis of the chosen model is still to be done, and so is its validation.
- They often leave several models that are nearly as good. Choose among them with what you know of the subject (which predictors should matter, which are easy to measure), prefer the simpler one, and say how you chose.

What the criteria do not do

- They rank models, they do not check them. The residual analysis of the chosen model is still to be done, and so is its validation.
- They often leave several models that are nearly as good. Choose among them with what you know of the subject (which predictors should matter, which are easy to measure), prefer the simpler one, and say how you chose.
- It uses the same response and the same cases. With missing values, R drops different rows for different models: remove the incomplete rows first.
- The model that wins was picked because it fits *these* data well: its p-values and its R^2 are somewhat optimistic.

What the criteria do not do

- They rank models, they do not check them. The residual analysis of the chosen model is still to be done, and so is its validation.
- They often leave several models that are nearly as good. Choose among them with what you know of the subject (which predictors should matter, which are easy to measure), prefer the simpler one, and say how you chose.
- It uses the same response and the same cases. With missing values, R drops different rows for different models: remove the incomplete rows first.
- The model that wins was picked because it fits *these* data well: its p-values and its R^2 are somewhat optimistic.

In a report, "the software chose this model" is not a justification: give the criterion, the candidates, and the reason for the final choice!

Automatic search procedures

When there are too many models

With 30 candidate predictors there are 2^{30} models, about a billion. A search procedure moves through them one predictor at a time:

- Backward elimination: start from the model with all the predictors, then remove the predictor whose removal improves the criterion most and stop when no removal improves it.
- Forward selection: start from the intercept alone, then add the predictor that improves the criterion most and stop when no addition improves it.
- Stepwise regression: forward, but after each addition, check whether a predictor already in can now be removed.

When there are too many models

With 30 candidate predictors there are 2^{30} models, about a billion. A search procedure moves through them one predictor at a time:

- Backward elimination: start from the model with all the predictors, then remove the predictor whose removal improves the criterion most and stop when no removal improves it.
- Forward selection: start from the intercept alone, then add the predictor that improves the criterion most and stop when no addition improves it.
- Stepwise regression: forward, but after each addition, check whether a predictor already in can now be removed.

The criterion can be a test (enter or stay if the p-value of the t test is below a threshold, often 0.10 to 0.20). R's `step()` uses the AIC.

Backward elimination with `step()`

```
full <- lm(log(Y) ~ X1 + X2 + X3 + X4 + X5 + X6 +
             X7 + X8, data = surg)
back <- step(full, direction = "backward", trace = 0)
back$anova
```

```
  Step Df    Deviance Resid. Df Resid. Dev       AIC
1      NA          NA        45   1.970510 -160.7773
2 - X4  1 0.001262737        46   1.971773 -162.7428
3 - X7  1 0.032562516        47   2.004335 -163.8583
```

Backward elimination with `step()`

```
full <- lm(log(Y) ~ X1 + X2 + X3 + X4 + X5 + X6 +
             X7 + X8, data = surg)
back <- step(full, direction = "backward", trace = 0)
back$anova
```

```
  Step Df    Deviance Resid. Df Resid. Dev       AIC
1      NA          NA        45   1.970510 -160.7773
2 - X4  1 0.001262737        46   1.971773 -162.7428
3 - X7  1 0.032562516        47   2.004335 -163.8583
```

`trace = 0` turns off the table that `step()` prints at each step; `back$anova` keeps the path. `Resid. Dev` is SSE_p. X_4 goes first, then X_7; any other removal would raise the AIC. The search stops at $X_1, X_2, X_3, X_5, X_6, X_8$: the smallest AIC of the table.

Backward elimination with `step()`

```
full <- lm(log(Y) ~ X1 + X2 + X3 + X4 + X5 + X6 +
             X7 + X8, data = surg)
back <- step(full, direction = "backward", trace = 0)
back$anova
```

```
  Step Df    Deviance Resid. Df Resid. Dev       AIC
1      NA          NA        45   1.970510 -160.7773
2 - X4  1 0.001262737        46   1.971773 -162.7428
3 - X7  1 0.032562516        47   2.004335 -163.8583
```

`trace = 0` turns off the table that `step()` prints at each step; `back$anova` keeps the path. `Resid. Dev` is SSE_p. X_4 goes first, then X_7; any other removal would raise the AIC. The search stops at $X_1, X_2, X_3, X_5, X_6, X_8$: the smallest AIC of the table.

With `k = log(n)` in `step()`, the criterion is the BIC: two more predictors go, and it stops at X_1, X_2, X_3, X_8.

Forward selection and stepwise regression

```
null <- lm(log(Y) ~ 1, data = surg)
forw <- step(null, scope = formula(full),
             direction = "forward", trace = 0)
```

Forward selection and stepwise regression

```
null <- lm(log(Y) ~ 1, data = surg)
forw <- step(null, scope = formula(full),
             direction = "forward", trace = 0)
```

Forward selection adds $X_3, X_2, X_8, X_1, X_6, X_5$, in this order, and stops at the same model. With `direction = "both"`, the stepwise search takes the same path here.

Forward selection and stepwise regression

```
null <- lm(log(Y) ~ 1, data = surg)
forw <- step(null, scope = formula(full),
             direction = "forward", trace = 0)
```

Forward selection adds $X_3, X_2, X_8, X_1, X_6, X_5$, in this order, and stops at the same model. With `direction = "both"`, the stepwise search takes the same path here.

- A search follows one path: it can miss the best model, and the three procedures can disagree.
- A `factor` column enters or leaves with all its indicators. Here X_7 and X_8 are two separate columns, so X_7 could go alone: the model then compares severe use with the rest.
- It is a way to find candidates, not a way to choose for you.

Model validation

Why validate?

The chosen model was picked because it fits *these* data well: its MSE and its R^2 flatter it.

Why validate?

The chosen model was picked because it fits *these* data well: its MSE and its R^2 flatter it.

Validation: check how the model does on data that were used neither to choose it nor to fit it.

- Collect new data, and predict them.
- Compare the coefficients with theory, or with earlier studies.
- Split the data: a training set to choose and fit the model, and a validation set (or test set), kept aside from the start.

Why validate?

The chosen model was picked because it fits *these* data well: its MSE and its R^2 flatter it.

Validation: check how the model does on data that were used neither to choose it nor to fit it.

- Collect new data, and predict them.
- Compare the coefficients with theory, or with earlier studies.
- Split the data: a training set to choose and fit the model, and a validation set (or test set), kept aside from the start.

The surgical unit: 108 patients, split at random. The 54 of Friday are the training set; the other 54 were kept aside.

The mean squared prediction error

Fit the model on the training set. For each of the n^* cases of the validation set, predict Y_i with the training coefficients:

$$MSPE = \frac{1}{n^*} \sum_{i=1}^{n^*} \left(Y_i - \hat{Y}_i\right)^2.$$

The mean squared prediction error

Fit the model on the training set. For each of the n^* cases of the validation set, predict Y_i with the training coefficients:

$$MSPE = \frac{1}{n^*} \sum_{i=1}^{n^*} \left(Y_i - \hat{Y}_i\right)^2.$$

- Divide by n^*, not $n^* - p$: no parameter is estimated on the validation set.
- Compare with the MSE of the training fit. If the $MSPE$ is close to it, the MSE is a fair measure of how well the model predicts. If it is much larger, the model fits the training set better than it predicts new cases.

The mean squared prediction error

Fit the model on the training set. For each of the n^* cases of the validation set, predict Y_i with the training coefficients:

$$MSPE = \frac{1}{n^*} \sum_{i=1}^{n^*} \left(Y_i - \hat{Y}_i\right)^2.$$

- Divide by n^*, not $n^* - p$: no parameter is estimated on the validation set.
- Compare with the MSE of the training fit. If the $MSPE$ is close to it, the MSE is a fair measure of how well the model predicts. If it is much larger, the model fits the training set better than it predicts new cases.

Once the model is validated, refit it on all the data for the final estimates.

Exercise: the 54 patients kept aside

The model with X_1, X_2, X_3, X_8 (chosen by BIC and PRESS) is fitted on the 54 training patients: $SSE = 2.1778$, with $p = 5$.

It then predicts the 54 patients kept aside: $\sum_{i=1}^{54}(Y_i - \hat{Y}_i)^2 = 4.174$.

1. Compute the MSE of the training fit, and the $MSPE$.
2. Does the model predict new patients as well as its MSE suggests?

Exercise: solution, and the three candidates

$MSE = 2.1778/49 = 0.0444$ and $MSPE = 4.174/54 = 0.0773$. The same for Friday's other two candidates, MSE on the training set and $MSPE$ on the validation set:

Predictors	Chosen by	p	MSE	$MSPE$
X_1, X_2, X_3, X_8	BIC, PRESS	5	0.0444	0.0773
X_1, X_2, X_3, X_6, X_8	C_p	6	0.0434	0.0764
$X_1, X_2, X_3, X_5, X_6, X_8$	R^2_a, AIC	7	0.0426	0.0794

Exercise: solution, and the three candidates

$MSE = 2.1778/49 = 0.0444$ and $MSPE = 4.174/54 = 0.0773$. The same for Friday's other two candidates, MSE on the training set and $MSPE$ on the validation set:

Predictors	Chosen by	p	MSE	$MSPE$
X_1, X_2, X_3, X_8	BIC, PRESS	5	0.0444	0.0773
X_1, X_2, X_3, X_6, X_8	C_p	6	0.0434	0.0764
$X_1, X_2, X_3, X_5, X_6, X_8$	R^2_a, AIC	7	0.0426	0.0794

- The $MSPE$ is 1.7 to 1.9 times the MSE: predicting new patients is harder than fitting the old ones. It stays of the same order: the models are acceptable, with a prediction error near 0.08, not 0.04.
- The three predict equally well: the extra predictors buy nothing. Keep X_1, X_2, X_3, X_8.

Exercise: solution, and the three candidates

$MSE = 2.1778/49 = 0.0444$ and $MSPE = 4.174/54 = 0.0773$. The same for Friday's other two candidates, MSE on the training set and $MSPE$ on the validation set:

Predictors	Chosen by	p	MSE	$MSPE$
X_1, X_2, X_3, X_8	BIC, PRESS	5	0.0444	0.0773
X_1, X_2, X_3, X_6, X_8	C_p	6	0.0434	0.0764
$X_1, X_2, X_3, X_5, X_6, X_8$	R^2_a, AIC	7	0.0426	0.0794

- The $MSPE$ is 1.7 to 1.9 times the MSE: predicting new patients is harder than fitting the old ones. It stays of the same order: the models are acceptable, with a prediction error near 0.08, not 0.04.
- The three predict equally well: the extra predictors buy nothing. Keep X_1, X_2, X_3, X_8.

Refitted on the new patients, the coefficient of age X_5 changes sign: one more reason to leave it out.

After validation

If we refit the same model on the validation set, and compare with the training fit:

- $MSPE$ close to the refitted MSE (here 0.0773 against 0.0775): the estimates from the training set predict the new cases as well as estimates computed from the new cases themselves. Only the training MSE was optimistic. Report the $MSPE$ as its prediction error.

Using the validation set well

- Choose among models on the training set only: with the criteria, or by cross-validation within it (slide 20).
- The validation set checks the final candidates, once. Each extra use to choose makes its $MSPE$ more optimistic: it becomes part of the training.

Using the validation set well

- Choose among models on the training set only: with the criteria, or by cross-validation within it (slide 20).
- The validation set checks the final candidates, once. Each extra use to choose makes its $MSPE$ more optimistic: it becomes part of the training.
- Once the model is validated, refit it on all the data (here 108 patients) for the final estimates.

A validation set in R

The acceleration data (390 cars) and Friday's three models for `acc`: 273 cars to train, 117 to validate.

```r
set.seed(1)
train <- sample(nrow(autompg), 273)
fit <- lm(acc ~ wt + log(hp), data = autompg[train, ])
pred <- predict(fit, newdata = autompg[-train, ])
mean((autompg$acc[-train] - pred)^2)          # MSPE
```

A validation set in R

The acceleration data (390 cars) and Friday's three models for `acc`: 273 cars to train, 117 to validate.

```
set.seed(1)
train <- sample(nrow(autompg), 273)
fit <- lm(acc ~ wt + log(hp), data = autompg[train, ])
pred <- predict(fit, newdata = autompg[-train, ])
mean((autompg$acc[-train] - pred)^2)          # MSPE
```

	set.seed(1)		set.seed(2)	
Model	MSE	$MSPE$	MSE	$MSPE$
`wt + hp`	3.33	2.50	2.90	3.49
`wt + log(hp)`	2.92	2.15	2.68	2.78
`wt + hp + I(hp^2)`	2.84	2.13	2.59	2.74

A validation set in R

The acceleration data (390 cars) and Friday's three models for `acc`: 273 cars to train, 117 to validate.

```
set.seed(1)
train <- sample(nrow(autompg), 273)
fit <- lm(acc ~ wt + log(hp), data = autompg[train, ])
pred <- predict(fit, newdata = autompg[-train, ])
mean((autompg$acc[-train] - pred)^2)          # MSPE
```

	`set.seed(1)`		`set.seed(2)`	
Model	MSE	$MSPE$	MSE	$MSPE$
`wt + hp`	3.33	2.50	2.90	3.49
`wt + log(hp)`	2.92	2.15	2.68	2.78
`wt + hp + I(hp^2)`	2.84	2.13	2.59	2.74

The ranking holds, but the $MSPE$ depends on which cars fell in the validation set: below the MSE with one split, above it with the other. The $MSPE$ is itself random: to stabilize it, repeat the split and average. Done systematically, this is k-fold cross-validation.

k-fold cross-validation

When there are too few cases to keep a validation set aside, reuse them all:

1. Split the cases at random into k folds of about equal size (often $k = 5$ or 10).
2. For each fold: fit the model on the other $k - 1$ folds, and predict the cases of the fold.
3. CV_k is the average of the n squared prediction errors.

k-fold cross-validation

When there are too few cases to keep a validation set aside, reuse them all:

1. Split the cases at random into k folds of about equal size (often $k = 5$ or 10).
2. For each fold: fit the model on the other $k - 1$ folds, and predict the cases of the fold.
3. CV_k is the average of the n squared prediction errors.

Every case is predicted once, by a fit that did not use it, and helps to fit the model $k - 1$ times. Averaging over k fits makes CV_k much less dependent on the split than one validation set.

k-fold cross-validation

When there are too few cases to keep a validation set aside, reuse them all:

1. Split the cases at random into k folds of about equal size (often $k = 5$ or 10).
2. For each fold: fit the model on the other $k - 1$ folds, and predict the cases of the fold.
3. CV_k is the average of the n squared prediction errors.

Every case is predicted once, by a fit that did not use it, and helps to fit the model $k - 1$ times. Averaging over k fits makes CV_k much less dependent on the split than one validation set.

With $k = n$ (leave-one-out), each fold is one case: $CV_n = PRESS_p/n$, Friday's criterion, and the n fits are not needed.

Cross-validation in R

```r
k <- 10; set.seed(1)
fold <- sample(rep(1:k, length.out = nrow(autompg)))
e <- numeric(nrow(autompg))
for (j in 1:k) {
  out <- fold == j                          # the cars of fold j
  fit <- lm(acc ~ wt + log(hp), data = autompg[!out, ])
  e[out] <- autompg$acc[out] - predict(fit, autompg[out, ])
}
mean(e^2)                                   # CV_10
```

Cross-validation in R

```r
k <- 10; set.seed(1)
fold <- sample(rep(1:k, length.out = nrow(autompg)))
e <- numeric(nrow(autompg))
for (j in 1:k) {
  out <- fold == j                          # the cars of fold j
  fit <- lm(acc ~ wt + log(hp), data = autompg[!out, ])
  e[out] <- autompg$acc[out] - predict(fit, autompg[out, ])
}
mean(e^2)                                   # CV_10
```

		CV_{10}		
Model	MSE	seed 1	seed 2	$PRESS_p/n$
wt + hp	3.03	3.07	3.12	3.08
wt + log(hp)	2.65	2.67	2.71	2.68
wt + hp + I(hp^2)	2.58	2.62	2.64	2.62

Cross-validation in R

```r
k <- 10; set.seed(1)
fold <- sample(rep(1:k, length.out = nrow(autompg)))
e <- numeric(nrow(autompg))
for (j in 1:k) {
  out <- fold == j                          # the cars of fold j
  fit <- lm(acc ~ wt + log(hp), data = autompg[!out, ])
  e[out] <- autompg$acc[out] - predict(fit, autompg[out, ])
}
mean(e^2)                                   # CV_10
```

		CV_{10}		
Model	MSE	seed 1	seed 2	$PRESS_p/n$
`wt + hp`	3.03	3.07	3.12	3.08
`wt + log(hp)`	2.65	2.67	2.71	2.68
`wt + hp + I(hp^2)`	2.58	2.62	2.64	2.62

`hp` alone predicts worst. `log(hp)` and the square predict about as well: the square slightly better inside the data, the log simpler to read and never turning. Either can be defended: say which, and why.